Berufsschule für Informationstechnik
Am Beruflichen Schulzentrum für Elektrotechnik
Strehlener Platz 2, 01219 Dresden

Ablaufdokumentation

Version 1.0

IT20/2
Lernfeld 9

Paul Görtler
Vincent Jablonski
Marcus Böhme

26. Februar 2022

Inhaltsverzeichnis

1 Firewall

1.1 Server aufsetzen

Für das im Projekt benötigte Firewall-System wird die Open-Source Software Lösung IPFire verwendet.

1.2 Konfiguration

2 DNS und DHCP

2.1 Server aufsetzen

Für eine sinnvolle Verwendung der Dienste DNS und DHCP wird eine eigenes System verwendet. Das System läuft auf einer CentOS 8 Version, welche direkt für die Nutzung im Serverbereich angepasst ist. Für die umsetzung von DNS und DHCP wird die Software Lösung DNSMASQ verwendet.

2.2 DNS-Service aufsetzen

2.3 DNS-Service konfigurieren

2.4 DHCP-Service aufsetzen

2.5 DHCP-Service konfigurieren

3 Datenbank

3.1 Server aufsetzen

Für eine sinnvolle Verwendung der Dienste DNS und DHCP wird eine eigenes System verwendet. Das System läuft auf einer CentOS 8 Version, welche direkt für die Nutzung im Serverbereich angepasst ist. Für die umsetzung der Datenbank wird die Software Lösung MariaDB verwendet.

3.1.1 Datenbank aufsetzen

Die nachfolgend beschriebenen Schritte zeigen den Prozess des Aufsetzens der Datenbank auf.

Für die Verwendung der Software MariaDB sind natürlich die erforderlichen Packages zu installieren. Mit dem folgenden Befehl kann das benötigte Package für einen MariaDB Server installiert werden.

```
1         $ dnf install mariadb-server wget unzip
```

Nach erfolgreicher Installation muss der Service aktiviert werden.

```
1         $ systemctl start mariadb
2         $ systemctl enable mariadb
```

3.2 Datenbank konfigurien

Im ersten Schritt wird die Konfiguration der MariaDB Datenbank gestartet.

```
1         $ mysql_secure_isntallation
```

Im laufe der Konfiguration werden Sicherheitstechnisch relevante Einstellungen getätigt. Die folgende Konfiguration wird verwendet.

Tabelle 1: MariaDB-Konfiguration

Einstellung	**Wert**
Root password?	rootPG9
Remove anonymous users?	Yes
Disallow root login remotely?	Yes
Remove test database and access to it?	Yes
Reload privilege tables now?	Yes

Als nächstes muss eine geeignete Datenbank mit einer Tabelle für die Speicherung der Tickets erstellt werden. Dieser Prozess ist anhand der folgenden Befehl ersichtlich.

```
$ mysql -u root -prootPG9
```

```
MariaDB [(none)]> CREATE DATABASE projekt3;
MariaDB [(none)]> use projekt3;
```

```
MariaDB [(projekt3)]> CREATE TABLE `tickets` (
    `id` int(11) NOT NULL,
    `title` char(255) NOT NULL,
    `creator` char(255) NOT NULL,
    `date` date NOT NULL,
    `description` mediumtext NOT NULL,
    `category` char(255) NOT NULL,
    `status` char(255) NOT NULL
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
MariaDB [(projekt3)]> ALTER TABLE `tickets` ADD PRIMARY KEY (`id`);
```

```
MariaDB [(projekt3)]> ALTER TABLE `tickets`
                      MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```

3.3 Backend

Für die Kommunikation zwischen dem Ticketsystem-Webserver und dem Datenbank Server wurde eine eigene API für den Datenbank Server entwickelt. Die Entwicklung wurde mit NodeJS und ExpressJS umgesetzt. Der Datenbank Server bekommt alle Datenobjekte vom Webserver zugesendet und kann diese direkt selbst abarbeiten und ggf. in die Datenbank aufnehmen. Der Datenbank Server ist öffentlich nicht zugänglich und demnach auch gegen Zugriffe von außen geschützt.

Listing 1: Datenbank API-Endpunkt: Ticket in Datenbank aufnehmen

```
// HANDLE THE POST OF THE TICKET DATA
app.post('/api/ticketsystem', (req, res) =>{
    var data = req.body
    var sqlConnection = mysql.createConnection({
        host: sqlSettings.host,
        user: sqlSettings.user,
        password: sqlSettings.password,
        database: sqlSettings.database });

    console.log(data)

    sqlConnection.connect(function(err) {
        sqlConnection.query(
        `INSERT INTO tickets (title, creator, date, description,
                              category, status)

        VALUES ('${data.title}', '${data.creator}', '${data.date}',
        '${data.desc}', '${data.category}', '${data.status}')`,
        function (err, result, fields){
            console.log(result);
        })
    })
    res.sendStatus(200)
})
```

4 Webserver

4.1 Server aufsetzen

Der Webserver läuft auf einem eigenem System, so wird ein sinnvolle Einbindung gewährleistet. Das System läuft auf einer CentOS 8 Version, welche direkt für die Nutzung im Serverbereich angepasst ist.

4.2 Backend

Der Webserver verfügt über ein selbst entwickeltes Backend. Die Entwicklung wurde mit NodeJS und ExpressJS umgesetzt. Das Backend ist für die Kommunikation zwischen dem Client-Frontend und der Ticketsystem-Datenbank zuständig. Der Webserver stellt das Ticketsystem-Frontend bereit.

Listing 2: Webserver API-Endpunkt: Ticket in Datenbank aufnehmen

```
// HANDLE THE POST OF THE TICKET DATA
// SEND THE PRIVIOUSLY COLLECTED DATA TO THE DATABASE SERVER'S API
app.post('/ticketsystem', (req, res) =>{
    res.sendStatus(200)

    var data = req.body
    console.log(data)

    try {
        const postReq = http.request(databaseOptions, res => {
            console.log(`statusCode: ${res.statusCode}`)

            res.on('data', d => {
                process.stdout.write(d)
            })

            res.on('error', d => {
                throw "connectionRefused_EXEPTION"
            })
        })

        postReq.write(JSON.stringify(data))
        postReq.end()

    } catch (e) {
        console.log('[ERROR]: Could not reach Database-Server')
    }
    res.end()
})
```

5 Ticketsystem

5.1 Frontend

Das Ticketsystem hat sein eigenes Frontend und ist so für jede Person nutzbar. Das Frontend basiert auf typischen HTML, CSS und JavaScript. Außerdem wird das Open-Source toolkit Bootstrap 5 als Framework für das Frontend verwendet. Bootstrap zeichnet sich durch seine schnelle Umsetzbarkeit von responsive Webseiten aus, welche direkt auf mehreren Endgeräten unterschiedlichster Art verwendet werden können.

5.2 Backend

Das Frontend muss natürlich mit einem passendem Backend ergänzt werden. Für die Entwicklung des Backends wird dem weltweitem Standard entsprechend JavaScript verwendet.

Das Backend ist grundlegend für die Kommunikation zwischen Client und Webserver da. Der Benutzer kann im Frontend ein Ticket erstellen, die Daten dieses Tickets werden dann vom Backend erfasst und an das Webserver-Backend gesendet. Der Webserver sendet die Ticketdaten and die Datenbank weiter. Das JavaScript-Backend liefert keine Informationen über den Datenbank-Server.

Listing 3: Backend Funktion: submitTicket

```
1         function submitTicket(){
2             var xhr = new XMLHttpRequest();
3             var url = "/ticketsystem";
4             xhr.open("POST", url, true);
5             xhr.setRequestHeader("Content-Type", "application/json");
6
7             var data = JSON.stringify({
8             "title": document.getElementById('ticket-title').value,
9             "creator": document.getElementById('ticket-creator').value,
10            "date": document.getElementById('ticket-date').value,
11            "description": document.getElementById('ticket-desc').value,
12            "category": document.getElementById('ticket-category').value,
13            "status": document.getElementById('ticket-status').value
14            });
15            xhr.send(data);
16        }
```